J. Symbolic Computation (1990), **10**, 253–280

A Fast Planar Partition Algorithm, I

KETAN MULMULEY

The University of Chicago, Ryerson Hall, 1100 East 58th Street, Chicago, Illinois 60637, USA

(*Received* 24 *May* 1988)

In this paper we give a fast randomized algorithm for finding a partition of the plane induced by a given set of line segments. The planar partition problem is interesting because it has important practical implications, especially in computer graphics, which indeed served as an inspiration for our investigation. Our algorithm is ideally suited for a practical use because: (i) it is extremely simple and robust, and (ii) despite this simplicity (or rather because of it) the algorithm is optimal; its expected running time is $O(m+n \log n)$, where n is the number of input segments and m is the number of points of intersection. The storage requirement is $O(m+n)$. Though the algorithm itself is simple, the global evolution of the underlying partition is non-trivial, which makes the analysis of the algorithm theoretically interesting in its own right.

1. Introduction

In this paper we give a fast and efficient randomized algorithm for finding a partition of the plane induced by a given set of line segments. The planar partition problem is interesting because it has important practical implications, especially in computer graphics, which indeed served as a starting point for our investigation. The algorithm which is used in practice for the related problems is the scanline algorithm. A primary motivation for this work was to come up with a practical alternative to the scanline algorithm which is (1) more efficient, (2) numerically more stable, (3) easily parallelizable, and (4) which can benefit significantly from a modest customized hardware. Our algorithm is a good candidate for two reasons: it is extremely simple to implement, and despite this simplicity (or rather because of) it is very efficient. Indeed the expected running time of our algorithm is optimal $O(m+n \log n)$, where n is the number of input segments and m is the number of points of intersection. Actually, we shall give two algorithms for finding the planar partition. The expected running time of the first algorithm is "practically" $O(m+n \log n)$, but this is not guaranteed. The expected running time of the second algorithm is provably $O(m+n \log n)$. The first algorithm is expected to perform better than the second one in practice. The memory requirement of both algorithms is $O(m+n)$. The analysis of the algorithms is theoretically interesting in its own right. Though the algorithms are very simple, the global evolution of the underlying partition is non-trivial, and the analysis must cope up with this non-trivial phenomenon. Like Quicksort, randomization is essential to the efficiency of the algorithms, because their worst case complexity is $O(n^2\alpha(n))$, where α is the inverse of Ackerman's function.

Independently, Chazelle & Edelsbrunner (1988), and Clarkson (1988) too have given optimal algorithms for the same problem, the former one being deterministic. Note that

An extended abstract of this paper appeared in the proceedings of the 29th Annual Symposium on Foundations of Computer Science, 1988.

0747-7171/90/090253+28 $03.00/0

the planar partition algorithm automatically gives all points of intersection of a given set of input segments. This intersection problem has been studied extensively before: Bentley & Oltmann (1979) presents an $O((m+n)\log n)$ scanline algorithm, Chazelle (1986) gives an $O(m+n\log^2 n/\log\log n)$ algorithm, and Edelsbrunner *et al.* (1986) gives an optimal, incremental algorithm for a special case when all segments are infinite lines. Our algorithm is randomized and incremental, and thus clearly demonstrates the power of combining the incremental strategy with randomization. In Mulmuley (1989*b*), we have recently given an efficient algorithm for the problem of hidden surface removal in computer graphics. It uses some of the ideas employed in the planar partition algorithm of this paper. We have also extended the planar partition algorithm to the case when the segments are parts of algebraic curves of bounded degree (Mulmuley, 1989*a*). Finally, it is possible to exploit the clustering which is often found in the input, especially the ones arising in computer graphics. In this paper, however, we make no assumption about the spatial distribution of the input segments.

2. The Planar Partition Problem

Let us first define the problem of planar partition. This is an extension of the segment intersection problem, previously studied in computational geometry. However, it is very convenient in applications. Suppose we are given a set of n segments in the plane. Each segment will be specified by its endpoints. We shall make no assumption about the spatial distribution of these segments: their lengths, orientations, as well as distribution can be arbitrary. The segments partition the plane into many regions. The resulting partition, however, has a disadvantage that its faces are not simple. Hence we prefer to work with a refinement of this partition which is convex. This partition is obtained as follows. Without loss of generality, assume that these segments are surrounded by a window. Pass through every endpoint a vertical segment which extends in upward (and downward) direction until it meets either the window border or one of the input segments. This induces a convex partition of the window. Our algorithm will construct this well defined partition. Figure 1(a) shows one input set of segments. Figure 1(b) shows the induced partition which the algorithm will find. In Figure 1(b), the vertical segment through the endpoint 10 meets the window border at 10′ above and another input segment at 10″ below. The points such as 10′ and 10″ will be called points of attachments. A point such as v will be called a point of intersection. One of the jobs of the algorithm is to find the points of intersection and attachments. By a vertex of a partition we shall mean (1) an endpoint of an input segment, (2) a point of intersection, (3) a point of attachment or (4) a corner of the window. Such a partition is traditionally specified by the obvious planar graph formed on these vertices. However, our algorithm will use a different representation. Towards this end, let us make the following

DEFINITION 1. A vertex v of the partition is said to be visible in the face R if ∂R, the boundary of R, has a tangent discontinuity at v.

In the partition given in Figure 2, l is visible in R_6 and R_3 but it is invisible in R_5. The vertex g is visible in R_2, R_6, R_4, R_5.

Each face of the partition will be specified by the list of *visible* vertices on its border linked in the counter-clockwise order. (We do not need double links between consecutive vertices on the border, which results in a substantial saving of the memory. This will become clear, if the reader goes through our algorithm carefully.) The length of the above

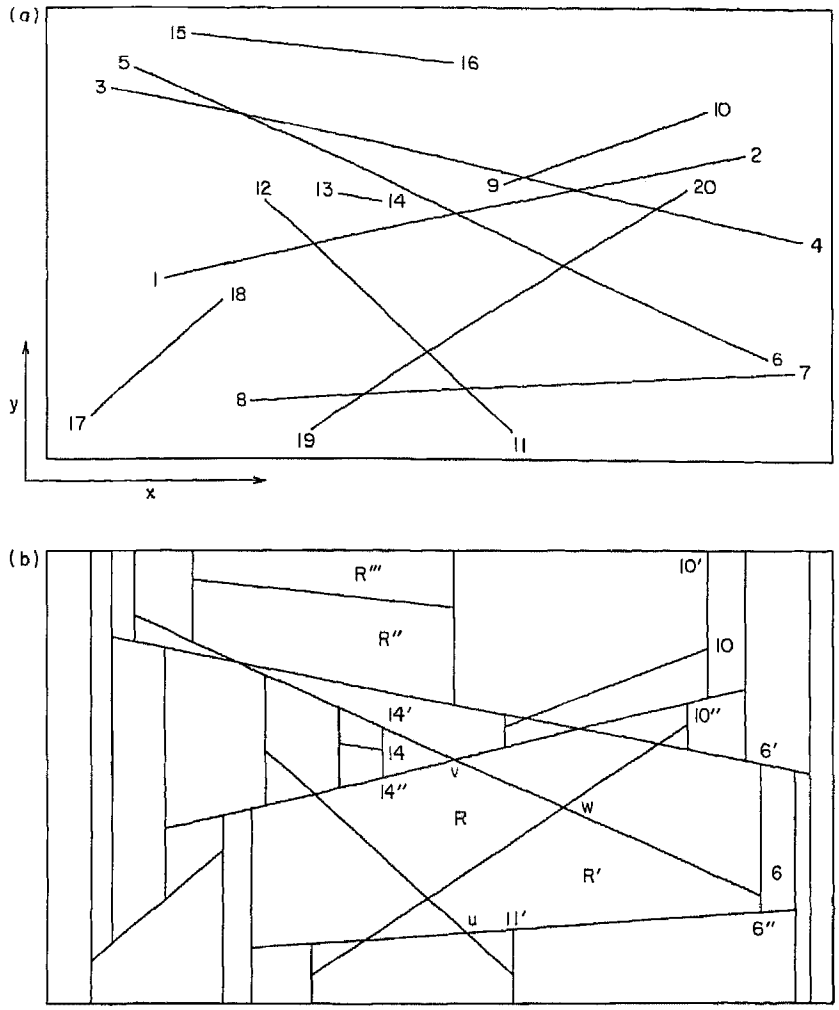


Figure 1. (a) The input segments; (b) the partition found by the algorithm; (c) the initial partition G_0; (d) the partition G_4 obtained after adding four randomly selected segments; (e) R_0 traversal; (f) R_0 to R_1 transition; (g) R_1 to R_2 transition; (h) the partition of G_5; (i) the partition G_4' produced by the second algorithm after adding four randomly selected segments; (j) the partition G_5' produced by the second algorithm after adding the fifth segment; (k) an alternate representation of G_4'.

mentioned list is equal to the number of visible vertices on the border of the face, which we shall call the facelength. If v is visible in R we shall refer to the corresponding entry in the representation of R by v_R. In addition to specifying the faces of the partition, we also need to specify the adjacency relationship at each vertex to complete the representation. For every vertex v, we shall link in the counter-clockwise order the entries v_R, for every R in which v is visible (see Figure 2). For a vertex v visible in R, let *successor*(v) denote the next visible vertex on ∂R in the counter-clockwise order. Given the entry v_R for a vertex v in the representation of the face R, let *neighbour*(v_R) denote the entry for v in the next face, according to the counter-clockwise order, in which v is visible. For example, in Figure 2 *successor*$(g_{R_5}) = j_{R_5}$, *successor*$(h_{R_5}) = g_{R_5}$, *neighbour*$(l_{R_3}) = l_{R_6}$, *neighbour*$(l_{R_6}) = l_{R_3}$, *neighbour*$(g_{R_4}) = g_{R_2}$, *neighbour*$(g_{R_2}) = g_{R_6}$, and so on.

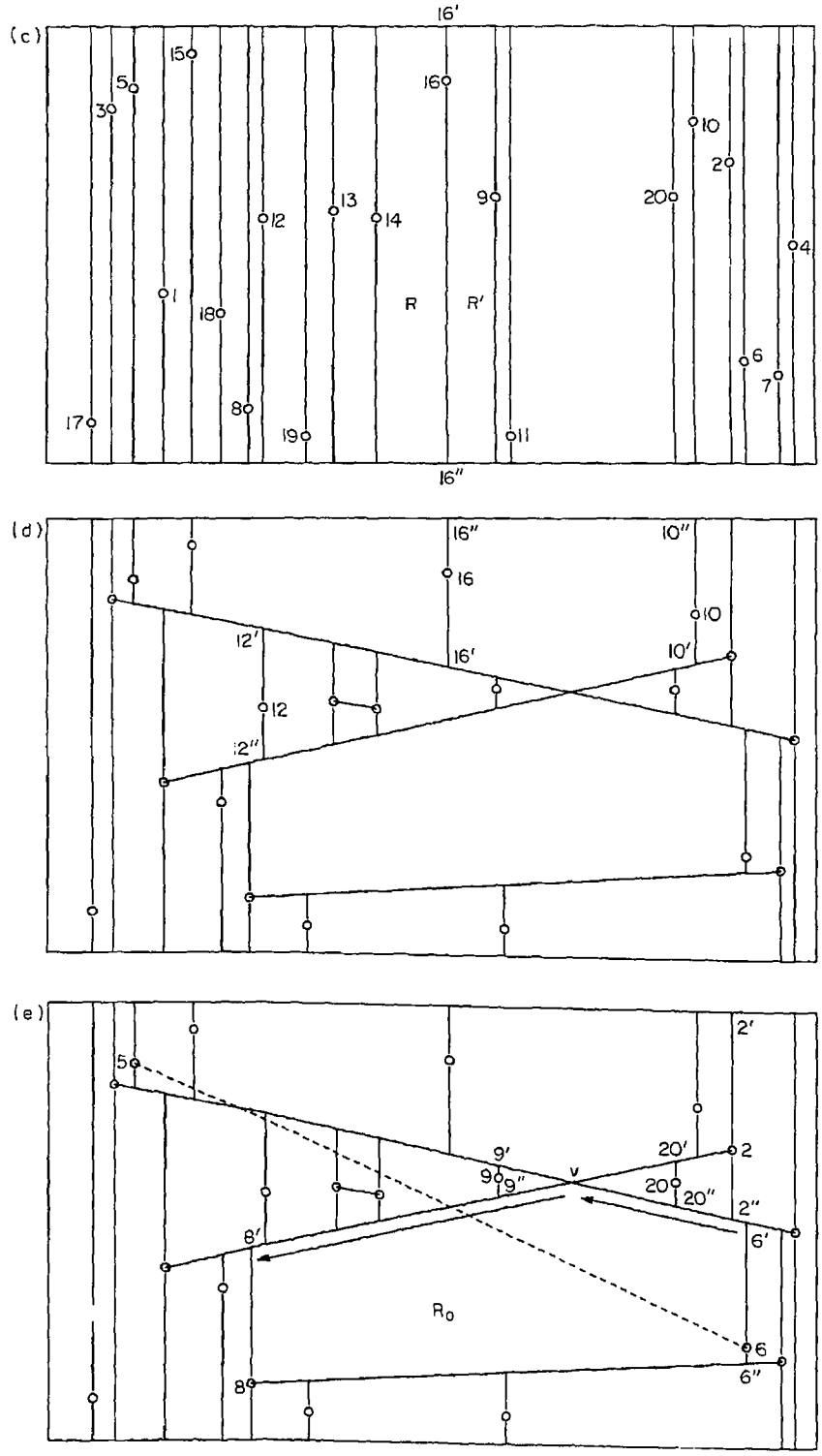


Figure 1—continued

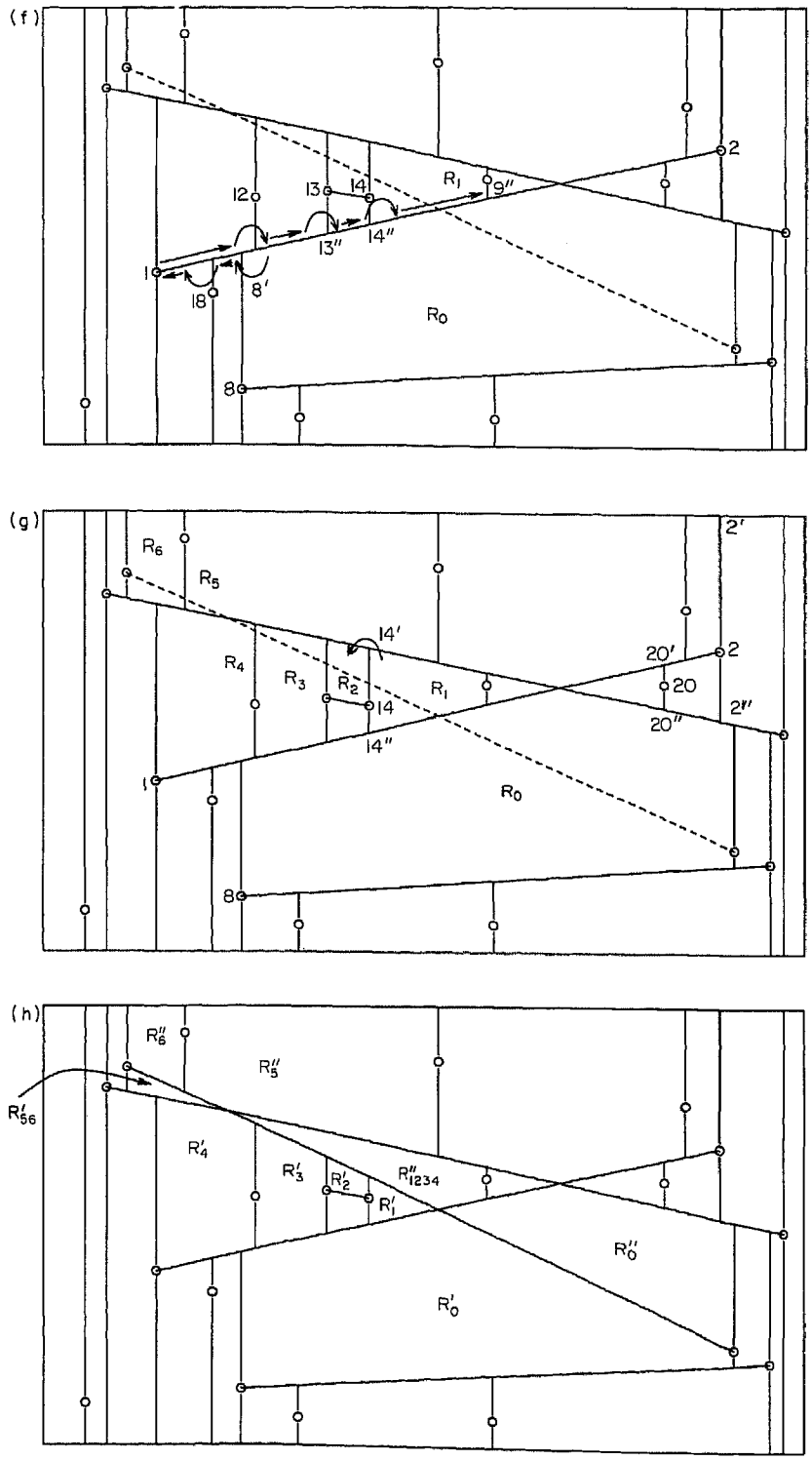


Figure 1—continued

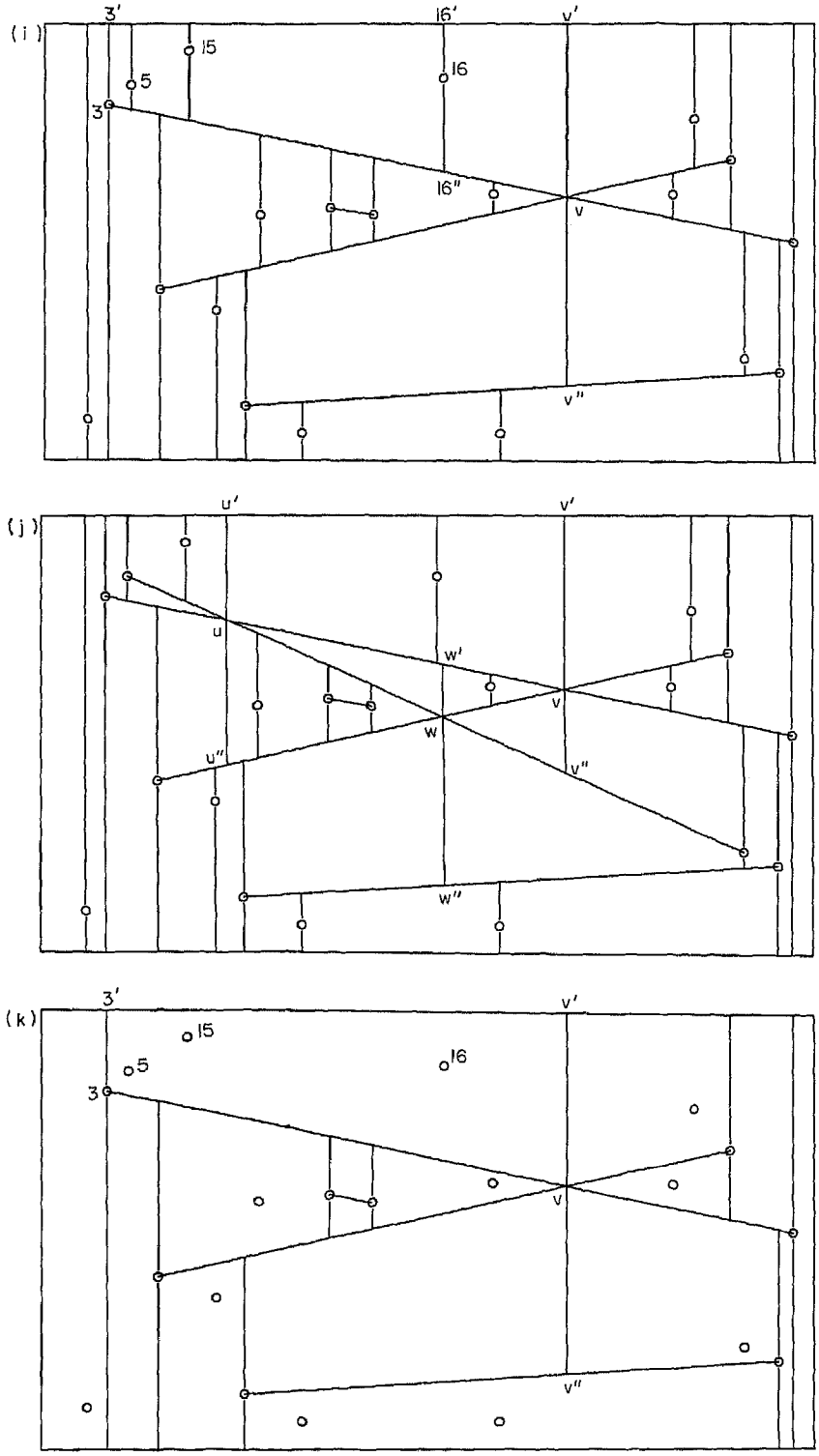
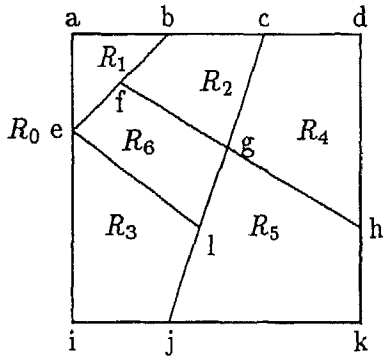


Figure 1—continued



R_0:	(a, i, k, d)	$a: a_{R_1} \to a_{R_0} \to a_{R_1}$
R_1:	(a, e, b)	$b: b_{R_1} \to b_{R_2} \to b_{R_1}$
R_2:	(f, g, c, b)	$e: e_{R_1} \to e_{R_3} \to e_{R_6} \to e_{R_1}$
R_3:	(e, i, j, l)	$l: l_{R_6} \to l_{R_3} \to l_{R_6}$
R_4:	(c, g, h, d)	$g: g_{R_2} \to g_{R_6} \to g_{R_5} \to g_{R_4} \to g_{R_2}$
R_5:	(j, k, h, g)	
R_6:	(e, l, g, f)	
Face representations		Adjacencies at some vertices

Figure 2. An example of a partition.

A border of the face R is defined to be the segment of ∂R between two consecutive *visible* vertices on ∂R. The counter-clockwise orientation of a face R also orients its borders. In the given representation of R, a border is not represented explicitly, but rather it is uniquely identified with its tail; using the successor relationship, the head of the border is obtained immediately. Henceforth we shall freely refer to borders in the description of the algorithm, keeping this identification in mind. A border of R having u as its tail and v as its head will be denoted by $(u, v)_R$ or (u_R, v_R) for the sake of non-ambiguity. In Figure 2, (g, j) is a border of R_5, which can be denoted by either $(g, j)_{R_5}$ or (g_{R_5}, j_{R_5}). Note that (g, l) is not a border of R_5.

3. A Planar Partition Algorithm

3.1. FIRST ALGORITHM

We shall now give a very efficient algorithm for finding the planar partition. For the sake of simplicity, we shall make the following non-degeneracy assumptions, which will be removed later. We shall assume that the x-coordinates of all endpoints are distinct. As a special case this rules out vertical input segments or two input segments sharing an endpoint. In addition, we shall assume that other degeneracies, such as three segments sharing a point of intersection, do not occur in the course of the algorithm.

Now we begin the description of the algorithm. Initially we pass through each endpoint in the input a vertical segment (also called a vertical attachment) which extends in either direction, until it hits the window border. This gives us a "stripped" initial partition G_0. For the input of Figure 1(a) the initial partition is shown in Figure 1(c). With each vertical

attachment we have also shown the endpoint associated with it. This association can be described precisely as follows. Consider the vertical segment $(16', 16'')$ through the endpoint 16. The vertical segment $(16', 16'')$ itself has no explicit representation in our representation of the partition. However, there are two borders $(16'', 16')_R$ and $(16', 16'')_{R'}$ in our partition which are adjacent with this vertical segment. We associate a pointer with each of these borders to the endpoint 16. In the reverse direction, there will be pointer from the endpoint 16 to either $(16'', 16')_R$ or $(16', 16'')_{R'}$. Which border is chosen is purely a matter of convention, because given $(16'', 16')_R$ we can immediately access $(16', 16'')_{R'}$ using the adjacency at 16", and conversely. During the course of the algorithm the segment $(16', 16'')$ is going to "contract" (or shrink). Whenever this happens, we assume that the pointers above are updated appropriately. Thus, given any endpoint we can immediately locate it in the partition and conversely given a vertical border we immediately know the associated endpoint. Though a vertical segment such as $(16', 16'')$ has no explicit representation in our data structure, we shall refer freely to the vertical segments in the description of the algorithm, with the tacit understanding that such a description can be immediately translated in terms of the adjacent vertical borders.

In the outline, the algorithm is as follows. Once G_0 is formed, we *randomly* select an input segment and add it to the partition to get a refined partition G_1, then select yet another input segment *randomly* to get the next refined partition G_2, and then yet another ..., until we are done. Before we describe what is the nature of the refinement and how it is done, let us state an invariant which each G_k will satisfy:

CONTRACTION INVARIANT. Assume that G_k was obtained by adding the randomly selected input segments $S_1, \ldots, S_k$ to the partition. Take any endpoint v in the input. This need not be an endpoint of one of the S_i, it could as well be an endpoint of an input segment not yet added. The vertical attachment through v will extend in G_k upwards and downwards until it meets either the window border or one of the S_i, $1 \leq i \leq k$. Thus the initial vertical attachment (in G_0) through each v has been contracted (or shrunk).

Given $S_1, \ldots, S_k$ (the order is not important), the contraction invariant uniquely defines G_k. It also implies that each face of G_k is simple (i.e. it has no holes) and convex. (Formally, simplicity follows from convexity.) In Figure 1(d) we have shown G_4 obtained after adding four randomly selected segments from the input shown in Figure 1(a). Notice how the vertical attachments such as $(16', 16'')$ and $(12', 12'')$ have shrunk. For this reason the vertical attachments will be called contractible.

Inductively assume that G_k is given to us. Let $S = S_{k+1}$ be the next randomly selected input segment. Now we shall describe how G_{k+1} is obtained. The procedure will be illustrated by showing how G_5 (Figure 1(h)) is obtained from G_4 (Figure 1(d)) when a randomly selected segment $S_5 = (6, 5)$ is added.

Let $S = S_{k+1} = (s_0, s_1)$ be the segment to be added; s_0 will be called the starting point. The procedure consists of (1) locating the starting point s_0 in G_k, (2) travelling through G_k from s_0 to s_1 along S, and (3) updating G_k along the way. The partition obtained at the end is G_{k+1}. We shall describe three steps in detail.

Locating the Starting Point

This is readily done by using the pointer from s_0 to the associated vertical attachment. By examining the two regions adjacent with this segment we know the first region R_0 that S begins to traverse. It is crucial here that the vertical attachment associated with s_0 existed in the partition even before S was added. The contractible vertical attachment

thus keeps track of the location of the endpoint associated with it in the evolving partition. As we shall see, it serves many other functions too.

Travelling Through the Partition

Once we know R_0 we traverse it to find the other point of intersection w of S with ∂R. This point is unique due to convexity of R_0. This step is called face traversal. If $w = s_1$, we are done. Else we find the next region R_1 that S emerges into; this step is called a face transition. We do face traversal and face transition repeatedly until we reach s_1. In the process we would have encountered many faces of G_k, say $R_0, \ldots, R_l$ in that order. This will be the usual and straightforward procedure of travelling through the partition, if it were not for the partial visibility of many vertices in the partition. Visibility is, however, an important notion in our algorithm, hence how it affects our travel is crucial. We shall describe the face traversal and the face transition in more detail. Let $F(x, y) = ax + by + c = 0$ be the linear equation satisfied by S. We assume that F is such that it assumes a positive value on our left side and a negative value on our right side as we travel from s_0 to s_1. Given any vertex v, we thus immediately know if it is to the left or to the right of the segment, or if it is on the segment.

(a) Face traversal. Suppose S starts traversing the face R at the border (v_R, w_R), where w_R is to the right of S. The following function returns the other (unique) border of R that S intersects.

function face-traversal (w_R)
  $h := w_R$;
  while *successor*(h) is to the right of S do $h =$ *successor*(h); return $(h,$ *successor*$(h))$;

The traversal of R_0 is shown in Figure 1(e). Here, given $6'_{R_0}$, the function will return $(v_{R_0}, 8'_{R_0})$. Note that in this traversal the invisible vertices 2″, 20″, 9″ etc. are skipped. This makes face traversal quite efficient. (Because, in the earlier half of the algorithm there will be typically many invisible vertices sitting on the border of any given face.)

(b) Face transition. Suppose we have finished traversing R_i and we know its border (u_{R_i}, v_{R_i}) through which S is about to exit. The vertex v_{R_i} is to the left of S. Let T be the segment containing the border (u_{R_i}, v_{R_i}). We want to find out the face R_{i+1} that S emerges into, on the other side of T. The following function returns the right vertex of the border of entrance in R_{i+1}. The idea is to travel on T to the left all the way until we reach the first vertex which is visible on both sides of T. Here we turn around, and travel to the right until we reach S again from the other side. When we do so, we are in R_{i+1}. See Figure 1(f), which shows R_0 to R_1 transition in our example.

function face-transition (v_{R_i})
  $h =$ *neighbour*(v_{R_i});
  while *successor*(h) is to the left of S do $h =$ *neighbour*(*successor* h);
return(*successor* h);

In Figure 1(f), given $8'_{R_0}$, the function will return $9''_{R_1}$. Note that the function successfully turns around at the vertex 1. The vertex where we turn around need not be an endpoint of some input segment as in Figure 1(f), it could also be a point of intersection. But we can not turn around at a point of attachment. In the above function, to find out if a vertex h on the segment T is to the left of the segment S, it is not necessary to evaluate

the linear form F at h. If we have already evaluated the coordinates of the point of intersection of S and T, comparing the x coordinate of this point with the x coordinate of h is enough. Other schemes are also possible.

If T, the segment containing the border $(u, v)_{R_i}$, were a vertical attachment, the above general procedure can still be followed. But there is a faster way which is guaranteed to take only constant time. Let w be the endpoint associated with this vertical border. The vertex v is assumed to be on the left side of S and the vertex u is on the right side. If w is to the left of S, *neighbour*(u) gives the required vertex in R_{l+1} (see Figure 1(g)). If w is to the right of S the situation is similar.

Updating the Partition

As one travels through G_k along S from s_0 to s_1 we encounter many faces, say $R_0, \ldots, R_l$, in that order. Obviously each R_i needs to be split. But if we did just the splitting, the algorithm will be quadratic. Besides the contraction invariant would not be maintained. The algorithm gains efficiency by suitably contracting every vertical attachment T in G_k that S intersects. If t, the endpoint associated with T, lies to the left of S we erase the right part T. If t lies to the right of S, the left part of T is erased. In either case, we are only retaining the interesting part of T (see Figure 1(h)). The contraction process causes many split halves of R_i faces to merge. For example in Figure 1(h), the face R''_{1234} was obtained by merging the split right halves $R''_1, R''_2, R''_3, R''_4$. Similarly R'_{56} was obtained by merging the split left halves R'_5 and R'_6. Conceptually G_{k+1} is obtained by first splitting R_0 to R_l and then merging the split halves as dictated by the contraction process. This is obviously not an efficient way to update G_k. One can simply assemble the new faces of G_{k+1} as we travel through G_k. At any stage there will be exactly one face on each side of S in the process of assembly. The obvious details are omitted.

Notice that G_{k+1} obtained at the end satisfies the contraction invariant. If there are n input segments, G_n obtained at the end is precisely the partition we sought.

In summary, there are three ingredients that are essential to the efficiency of the algorithm: (1) randomization, (2) the notion of vertex visibility, and (3) the notion of a contractible vertical attachment. Randomization helps the algorithm in the same way that it helps the sorting algorithm Quicksort. The contractible vertical attachment serves three purposes.

(1) It keeps track of the associated endpoint in the evolving partition, thereby making the location of the endpoint, when a segment is about to be added, easy.

(2) It keeps each face of G_k simple (i.e. it has no holes) and convex, thereby making the face traversal and the face split easy.

(3) In a subtle way, as we shall see in the analysis, it speeds up the face traversal and the face split even further, by keeping the average facelength (the average number of visible vertices on the border) small.

It is clear that the vertical attachment needs to be contractible to keep the number of face splits small. The notion of visibility mainly speeds up the face traversal and the face split by keeping the facelength small. Because of this notion one only needs to worry about the visible vertices on the border of the face during the face traversal and the face split. The same notion, on the other hand, entails an additional cost in the face transition. It is an important result from the analysis of this algorithm that this additional cost is small. Thus what is gained by speeding up the face traversals and the face splits, more than compensates for what is lost in the face transitions. In future, we shall refer to the combined step of face traversal and face split by simply a face split.

3.2. SECOND ALGORITHM

We shall now give a minor variant of the above algorithm, because it has a certain theoretical virtue, as will become clear in the next section. The variant is obtained as follows. Everytime the segment $S = S_{k+1}$ intersects S_i $(i \leq k)$, which is one of the segments already added, one passes through their point of intersection a *contractible* vertical attachment which extends in upward (and downward) direction until it meets the window border or one of the S_i, $i \leq k$. Being contractible, this vertical attachment contracts during the course of the algorithm just like any other contractible attachment. In Figure 1(i) we show the partition G_4' produced by this variant algorithm when we add the same four segments. Notice the vertical attachment through the vertex v. Figure 1(j) shows G_5' after adding $S_5 = (6, 5)$ as before. Notice the new vertical attachments through u and w. Also notice how the old attachment through v has shrunk.

3.3. A PRAGMATIC ISSUE

In the representation of the planar partition used so far, we pass vertical attachments through all endpoints in the given input, even the endpoints of the segments that are not yet added, at any given stage of the algorithm. Referring to Figure 1(i), we see that this has an effect of decomposing a trapezoid such as $33'v'v$ into a number of vertical strips. It is possible to avoid this unnecessary decomposition into vertical strips, and thus gain more efficiency, by using the following alternate representation. In this representation, we do not pass a vertical attachment through an endpoint of a segment, if the segment has not been added so far in the algorithm (see Figure 1(k)). Now with each trapezoid, such as $33'v'v$ in Figure 1(k), we associate a list of endpoints (of the unadded segments) that lie in the trapezoid. Furthermore this list is kept ordered by the x-coordinates of the endpoints. Thus with the trapezoid $33'v'v$ in Figure 1(k), we associate an ordered list (5, 15, 16). These lists have to be properly updated and split during the algorithm, as new segments are added. Because of our notion of visibility as in Definition 1, it is easy to see that this new representation is equivalent to the previous one. However, it has two advantages. Firstly, we do not need to maintain and calculate points of attachments, such as $16''$ in Figure 1(i), that belong to vertical attachments through the endpoints of the segments, that have not been added so far. Secondly, it is possible to modify the algorithm, in an obvious way, so that such points of attachment are not visited during the face transitions; e.g. $12''$ will not be visited in the R_0 to R_1 transition, shown in Figure 1(f).

For the sake of simplicity, we shall continue to work with the previous representation in the rest of the paper. Our analysis of the second algorithm will be trivially seen to be applicable to this alternate representation as well.

4. Analysis

Before proceeding with the analysis, let us prove one very general fact about partitions.

For any partition G, let $b(G)$ denote the total number of its borders and let $f(G)$ denote the total number of its faces. Let $\mu(G) = b(G)/f(G)$; this is the average face length. Let $\delta(G) = 4 - \mu(G)$ be the deviation of $\mu(G)$ from 4; the importance of the number 4 will become clear soon. Let us say that we want to add a new segment S to this partition G. We assume that the endpoints of the segment to be added lie on the

borders of G. Let us assume for a moment that S does not cut through the already existing vertices and also that none of the two endpoints of S coincides with an already existing vertex *from its visible side.* (For example, in Figure 2, if we add a segment which cuts through the face R_1, it is perfectly alright if its endpoint coincides with f.) Let us call the resulting partition G'.

LEMMA 1.

1. (a) *If* $\mu(G)=4$ *then* $\mu(G')=4$.
   (b) *If* $\mu(G)<4$ *then* $\mu(G)<\mu(G')<4$.
   (c) *If* $\mu(G)>4$ *then* $\mu(G)>\mu(G')>4$.
2. *If we keep on adding segments to* G *in any fashion* (*as long as they do not coincide with a visible vertex*) *then letting* G_k *denote the successive partitions, we have* $\mu(G_k)\to 4$.

PROOF. Let us introduce a new quantity $\Delta(G)=\delta(G)f(G)$. We shall first show that in the process of such addition $\Delta(G)$ remains invariant, i.e. $\Delta(G')=\Delta(G)$.

Assume that the total number of vertices on S in the resulting partition G' is r. That means S cuts G in $r-2$ vertices or equivalently S has $r-1$ "spans" $S_0,\ldots,S_{r-2}$. Let us consider the span S_0, the other cases being similar. Say its starting point is v and that it cuts through a face R of G. Let B_0 be the border of R containing v and let B_1 be the other border of R which the span intersects. Then it will cut it into two faces thus increasing the number of faces by 1. Similarly it is easily seen that the split has increased the number of borders by 4; the already existing borders B_0, B_1 are split into two pieces each and S_0 itself adds two extra borders. Notice that introduction of v only splits the border B_0 of R. It does *not* split the adjacent border on the "other" side; this is where the notion of visibility is critical. To summarize, each span contributes an increase of 4 in the number of borders and an increase of 1 in the number of faces. Thus $f(G')=f(G)+r-1$ and $b(G')=b(G)+4(r-1)$. Hence

$$\Delta(G')=\delta(G')f(G')=4f(G')-b(G')=4f(G)-b(G)=\delta(G)f(G)=\Delta(G),$$

and also $\delta(G')=\Delta(G)/f(G')=\delta(G)f(G)/f(G')$.

First of all this means that δ does not change the sign in this addition and secondly if $\delta(G)$ is zero so is $\delta(G')$. Thirdly it is clear that $f(G)/f(G')<1$ and hence if $\delta(G)\neq 0$ we have $|\delta(G')|<|\delta(G)|$. From this the first part of the lemma follows.

Finally if we successively add many segments to G, then letting G_k denote the kth partition, and applying the invariance of Δ as many times, we conclude that $\Delta(G_k)=\Delta(G)$; i.e. $\delta(G_k)=\delta(G)[f(G)/f(G_k)]$. As $k\to\infty$, $f(G_k)\to\infty$ and hence $\delta(G_k)\to 0$, i.e. $\mu(G_k)\to 4$.

REMARK. Notice that the above lemma does not need the partition to be convex.

What the above result says is that regardless of what partition one starts from and regardless of the way one adds segments (as long as the coincidences are legal), $\mu\to 4$. Though the above theorem guarantees the convergence of μ, the average face length, to 4, our algorithm requires something stronger (we also need to incorporate deletions). Let us call a partition G critical if $\mu(G)=4$. Let us start with a critical partition G, and add to it any number of segments. However this time we shall allow the endpoints of the segments to coincide with the already existing visible vertices and we shall also allow segments to cut through the already existing vertices; these include the intersection vertices too. We can also delete a segment as long as it does not cut through the vertices of the remaining partition and does not coincide at its endpoints with the vertices of the remaining

partition from their *visible* sides. Let us call the partition that results after any number such additions and deletions G'. Then

COROLLARY 1. $\mu(G') \leq 4$.

PROOF. We shall prove by induction on the number of additions and deletions. We shall also assume, without loss of generality, that when we add a segment we get only one "span", i.e. to say the segment does not cut through the already existing vertices; otherwise we split the segment into an appropriate number of spans and add one span at a time. Because of our notion of visibility this splitting is justified. For $G_0 = G$ the corollary obviously holds. Let G_k be the partition at the kth stage. Let G_{k+1} be a partition obtained by adding a segment S to G_k. Let G^{ε}_{k+1} be a partition obtained by adding to G_k a slightly perturbed segment S_{ε} such that none of its endpoints coincide with a visible vertex. Then it is clear that $f(G^{\varepsilon}_{k+1}) = f(G_{k+1})$ and $b(G^{\varepsilon}_{k+1} \geq b(G_{k+1})$.

Hence $\mu(G_{k+1}) \leq \mu(G^{\varepsilon}_{k+1})$. By the induction hypothesis, $\mu(G_k) \leq 4$. Applying the previous lemma we conclude that $\mu(G^{\varepsilon}_{k+1}) \leq 4$. Hence $\mu(G_{k+1}) \leq 4$. The deletions can be treated similarly.

THEOREM 1. *The average facelength of the planar partition remains* ≤ 4 *throughout the algorithm.*

PROOF. It is clear that the initial stripped partition G_0 is critical. It is easy to prove inductively that the conditions of Corollary 1 hold whenever a segment is added to the partition and whenever a segment is deleted from the partition (this happens whenever a vertical attachment contracts).

Let m be the number of intersections of input segments. Let n be the number of the endpoints of the input segments. In the non-degenerate case being considered in this section, n is twice the number of input segments. (In the degenerate case, when segments are allowed to share endpoints, n will be smaller by a factor roughly equal to the average vertex degree in the input. Hence, this kind of degeneracy is actually going to speed up the algorithm.) If t is an endpoint of some segment in the input, let n_t denote the number of segments which intersect the imaginary vertical line through t. Let us define the average span length, s, of the input by $s = (\sum_t n_t)/n$. In the worst case $s = O(n)$, but realistically $s = O(\sqrt{n})$, see Sutherland *et al.* (1974).

For the first algorithm we prove

THEOREM 2. *The expected number of face splits* $\leq m + 2n(\ln(1+s)+1)$.

There are two kinds of face transitions in the algorithm. A face transition can take place either across a vertical attachment or across an input segment. For example, the transition in Figure 1(g) is across a vertical attachment $(14', 14'')$, whereas the transition in Figure 1(f) is across the input segment $(1, 2)$. As we have seen, a transition across a vertical attachment is achieved in constant time. So we need to worry only about the face transitions across the input segments. Consider the transition across the input segment $(1, 2)$ in Figure 1f. Apart from the "turn around" point 1 all other points visited in the face transition are points of attachment. This is the case for every face transition. Thus we need to estimate the expected value of the total number of points of attachments

visited during the face transitions in the whole course of the algorithm. The same point of attachment can be visited many times in the algorithm, in that case each visit is to be counted.

THEOREM 3. *The expected number of points of attachment visited during face transitions in the whole course of the algorithm is* $O(n \log s)$. *The constant within O is small.*

Note that these two theorems still do not prove that the first algorithm is $O(m+n \log n)$ in the randomized sense. Remember that the cost of the facesplit is proportional to its facelength, i.e. the number of visible vertices on its border. Theorem 1 and Theorem 2, lead one to expect that expected cost of face splits in the whole course of the algorithm should be $O(m+n \log n)$, but this is not a theoretical guarantee. In practice this is not a problem. Indeed in the implementation of the algorithm, it was found that most of the faces which arise in the course of the algorithm have facelengths 3, 4, or 5, as Theorem 1 would lead one to expect.

For the second variant algorithm, this problem does not arise. Indeed every face formed in the course of the algorithm is either a trapezoid or a triangle, hence its facelength is trivially less than or equal to four. For the second algorithm the following two theorems hold.

THEOREM 4. *The expected number of face splits in the second algorithm is* $\leq 4m+2n(1+\ln s)$, *where m, n, s are defined as before.*

THEOREM 5. *The expected number of points of attachments visited during the face transitions in the whole course of the second algorithm is* $O(m+n \ln s)$.

As every face formed in this algorithm has facelength ≤ 4, it follows that the expected running time of this algorithm is $O(m+n \log n)$.

For further reference let us state the following useful lemma.

LEMMA 2. (*Restriction Argument*): *Given two sets A and* $B \subset A$, *let* $\Sigma(A)$ *and* $\Sigma(B)$ *denote the sets of permutations on A and B respectively. Suppose we are given a uniform probability distribution on the set* $\Sigma(A)$. *Regard permutations as sequences. For a given* $\sigma \in \Sigma(B)$, *let* $p(\sigma)$ *be the probability of choosing a sequence from* $\Sigma(A)$ *which contains* σ *as a subsequence. Then p gives a uniform probability distribution on* $\Sigma(B)$.

PROOF. Easy.

4.1. BASIC RECURRENCE RELATIONS

In this section we shall prove some recurrence relations which are fundamental to the analysis of the algorithms. The notations m and n in this, as well as in the next few sections, will denote quantities which are different from what they denoted in the previous sections. The meaning should be clear from the context.

LEMMA 3. (*Ratio Lemma*): *If* $g(0, n)=0$ *and* $g(m, n)=cm/(m+n)+1/(m+n)\sum_{j=0}^{m-1} g(j, n)$, *where* $c>0$ *is a function of n, then* $g(m, n) \leq c \ln(1+m/n)$.

REMARK. The crucial point here is not the logarithmic dependence of the bound on m—that is clear—but the logarithmic dependence on the *ratio* m/n. This is extremely important in the analysis.

PROOF. By linearity of the recurrence relation, it suffices to consider the case when $c=1$.

We prove by induction on m that $g(m, n) \le \ln(1+m/n)$. If $m=0$ this is clear. Otherwise, because ln is a concave increasing function in $[1, \infty)$, we get that

$$\begin{aligned} g(m,n) &= \left(\frac{m}{m+n}\right) + \frac{1}{m+n}\sum_{j=0}^{m-1} g(j,n) \\ &\le \left(\frac{m}{m+n}\right) + \frac{1}{m+n}\sum_{j=0}^{m-1} \ln\left(1+\frac{j}{n}\right) \le \frac{m}{m+n} + \frac{1}{m+n}\int_0^m \ln\left(1+\frac{x}{n}\right) dx \\ &= \frac{m}{m+n} + \frac{n}{m+n}\left[\left(1+\frac{x}{n}\right)\ln\left(1+\frac{x}{n}\right) - \left(1+\frac{x}{n}\right)\right]_0^m \\ &= \frac{m}{m+n} + \frac{n}{m+n}\left(\frac{m+n}{n}\ln\left(1+\frac{m}{n}\right) - \frac{m}{n}\right) = \ln\left(1+\frac{m}{n}\right). \end{aligned}$$

LEMMA 4. (*Unit Convergence Lemma*): *If* $f(m, n)$ *satisfies*

$$f(0,n)=0 \quad \text{and} \quad f(m,n) = \frac{\ln\left(\frac{m}{n}+1\right)}{\left(\frac{m}{n}+1\right)} + \frac{1}{m+n}\sum_{j=0}^{m-1} f(j,n),$$

then, for all $m \ge 0$ *and* $n>0$, $f(m, n) \le 1$ *and* $\lim_{m\to\infty} f(m, n) = 1$.

Thus the bound depends neither on m nor on n. It follows that, if $g(m, n)$ *satisfies, for some* $c, d \ge 0$ *that are purely functions of n,*

$$g(0,n)=0,$$

$$g(m,n) \le \frac{c\ln\left(\frac{m}{n}+1\right)+d}{\frac{m}{n}+1} + \frac{1}{m+n}\sum_{j=0}^{m-1} g(j,n),$$

then $g(m, n) \le c+d$.

PROOF. By unfolding the definition of f we get $f(m, n) = \sum_{i=0}^{m-1} \Delta_n(m, i)$, where, for a fixed n, Δ_n is defined by

$$\Delta_n(m,0) = \frac{\ln\left(\frac{m}{n}+1\right)}{\left(\frac{m}{n}+1\right)} \quad \text{and} \quad \Delta_n(m,l) = \frac{1}{m+n}\sum_{j=0}^{m-1} \Delta_n(j,l-1) \quad \text{for } l>0.$$

Inductively it is easy to prove that $\Delta_n(m, l) \approx [\ln^{l+1}(m/n+1)]/[(l+1)!(m/n+1)]$, using the following asymptotic estimate

$$\sum_{i=0}^{h-1} \frac{\ln^k\left(\frac{i}{n}+1\right)}{\frac{i}{n}+1} \approx \int_0^h \frac{\ln^k\left(\frac{x}{n}+1\right)}{\frac{x}{n}+1}\,dx = n \ln^{k+1}\left(\frac{h}{n}+1\right)\Big/(k+1) \quad \text{for } h, k>0.$$

Hence

$$f(m, n) \approx \sum_{i=0}^{m-1} \frac{\ln^{i+1}\left(\frac{m}{n}+1\right)}{(i+1)!\left(\frac{m}{n}+1\right)} \approx (e^{\ln(m/n+1)}-1)\Big/\left(\frac{m}{n}+1\right) = \frac{m}{m+n}.$$

Hence $\lim_{m\to\infty} f(m, n) = 1$. To prove $f(m, n) \leq 1$, we shall, by induction on m, prove that $f(m, n) \leq m/(m+n) = 1 - n/(m+n)$. Basis ($m = 0$) is satisfied and inductively

$$f(m, n) \leq \frac{\ln\frac{m+n}{n}}{\frac{m}{n}+1} + \frac{1}{m+n}\sum_{j=0}^{m-1}\left(1-\frac{n}{j+n}\right) \leq \frac{\ln\frac{m+n}{n}}{\frac{m}{n}+1} + \frac{1}{m+n}\int_0^m \left(1-\frac{n}{x+n}\right)dx$$

because $1-\frac{n}{x+n}$ is concave and increasing,

$$= \frac{\ln\frac{m+n}{n}}{\frac{m}{n}+1} + \frac{1}{m+n}[x - n\ln(x+n)]_0^m$$

$$= \frac{\ln\frac{m+n}{n}}{\frac{m}{n}+1} + \frac{1}{m+n}\left[m - n\ln\left(\frac{m+n}{n}\right)\right] = \frac{m}{m+n}.$$

4.2. A PROBABILISTIC EXPERIMENT

In this section we shall describe one probabilistic experiment and analyse it. Its relevance to the analysis of our algorithm will become clear later. Suppose we are given a linearly *ordered* set M containing m elements and another disjoint set N (possibly unordered) containing n elements. (The notations m and n in this subsection have nothing to do with the number of intersections or the number input vertices which they denoted in the previous sections.) We successively choose an element from the union $M \cup N$ elements *without replacement* until an element in N is drawn, at which point the experiment ends. As a convention, if N is empty, the experiment proceeds all the way until every element of M is drawn. Let $a_1, \ldots, a_k$ be the elements in M chosen (in that order) before an element in N was chosen. We define the following two notions.

We say that $b \in M$ is visible (at the end of the experiment) if $b < a_i$ for all i (again this has nothing to do with the notion of visibility of the previous section). Imagine the elements in M placed on the real line according to their order with an observer sitting at the origin. See Figure 3, where we have shown the elements $a_1, \ldots, a_6$ ($k=6$ here) which were chosen in that order before an element in N was chosen. In Figure 3, $b \in M$ is visible at the end of the experiment if b lies between o and a_5. Thus $b \in M$ is visible if it can be seen by the observer o without being obscured by any a_i chosen during the whole experiment. We say that a_i was observed (by the observer when it was chosen) if for every $j < i$, $a_i < a_j$. The idea is that the observer O could see a_i when it was chosen without it being obscured by any a_j chosen before. In Figure 3, a_1, a_2, and a_5 were observable, others were not.

Denote by V the number of visible elements in M at the end of experiment and by O the number of observed elements among those chosen from M. Denote by $\bar{V}(m, n)$ the expected value of V and by $\bar{O}(m, n)$ the expected value of O. In the analysis of the algorithm we shall need very precise bounds on $\bar{V}(m, n)$ and $\bar{O}(m, n)$.

For the time being consider the following simpler situation. Let us assume that both m and n are large and that m is far larger than n. In this case the probability of drawing an element from N—this event is to be considered a success—is $p = n/(m+n)$ and the probability of drawing an element from M is $q = m/(m+n)$. As m and n are both very large and $m \gg n$, we can assume that p and q are constants.

Consider the following continuous analog of the experiment. Consider a Bernoulli trial in which we toss a coin repeatedly. The probability of getting heads is p and the probability of getting tails is $q = 1-p$. If the toss is tails we choose a point randomly (with uniform distribution) from the unit interval (0, 1) and then proceed by tossing the coin again. We stop as soon as the toss is heads, and the experiment ends. We say that $y \in (0, 1)$ is visible from the origin at the end of the experiment if it is not obscured by a point chosen, i.e. there is no y' chosen during the course of the experiment such that $0 < y' < y$. Obviously the set of visible points form an interval of the form $(0, V]$, $0 < V < 1$. We want to find the expected value of V, $E(V)$.

Towards this end let us first find the expected value of $X = 1 - V$.

The probability of drawing n points from the unit interval before the experiment ends is $q^n p$ and the conditional probability that $X \leq x$, after choosing n points, i.e. each $x_i \in [1-x, 1]$, is x^n. Hence

$$p[X \leq x] = \sum_{n=0}^{\infty} q^n p x^n = p/(1-xq).$$

By taking derivative, the density function for X is $pq/(1-xq)^2$. Hence the expected value of X is

$$\begin{aligned} E(X) = \int_0^1 \frac{pq}{(1-xq)^2}\, x\, dx &= \frac{p}{q}\int_p^1 \frac{1-y}{y^2}\, dp \quad \text{where } y = 1-xq \\ &= \frac{p}{q}\left[-\frac{1}{y} - \ln y\right]_p^1 = \frac{p}{q}\left[-1 + \frac{1}{p} + \ln p\right] \\ &= \frac{p}{q}\left[\frac{q}{p} + \ln p\right] = 1 + \frac{p}{q}\ln p. \end{aligned}$$

Hence $E(V) = 1 - E(X) = p/q \ln 1/p$.

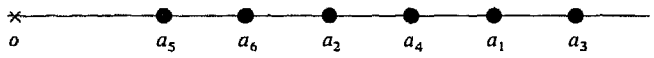


Figure 3

We can think of this "continuous" version as an approximation to our original experiment with $p = n/(m+n)$, and $q = m/(m+n)$, and with the unit interval corresponding to the set M after "normalizing" it. Hence to unnormalize we have to multiply $E(V)$ by m, and we conclude that the expected value of V in our original experiment is asymptotically given by

$$\bar{V}(m, n) \approx m\left(\frac{p}{q}\ln\frac{1}{p}\right) = n\ln\frac{m+n}{n} = n\ln\left(1+\frac{m}{n}\right).$$

Of course this is an asymptotic estimate and we also assumed that $m \gg n$. But in the analysis of the algorithm we need an estimate which works even when m and n are small and (or) comparable.

The following lemma says that the above estimate gives (a tight) upper bound for *all* m and n.

LEMMA 5. $\bar{V}(m, n) \leq n\ln(1+m/n)$ *for all* $n > 0$ *and* $m \geq 0$.

PROOF. The probability of choosing an element from N is $n/(m+n)$, and the expectation of V conditional on this event is m. The probability of choosing a fixed element c (which is $j+1$st in the ordering on M) is $1/(m+n)$. The expectation of V conditional on c being chosen is $\bar{V}(j, n)$. This needs some justification. Obviously after c is chosen the elements of M higher in ordering than c can be chosen during the course of the experiment. What we are saying in effect is that these choices can be ignored altogether. This is best seen as follows. Let M_j be the set of elements in M smaller than c. Imagine an observer who can only see the elements in M_j and N. Then as far as he can see, whenever an element in $M_j \cup N$ is chosen, it is still chosen with uniform probability. Hence, it is clear that the expectation of V conditional on c being chosen must depend only on M_j and N. The argument of this kind is going to be used so many times that we give it a name; we call it the restriction argument. Formally, it follows from Lemma 2. For example, in the present case, this will follow by letting $A = M \cup N$ and $B = M_j \cup N$ in that lemma. Thus we get the following recurrence:

$$\bar{V}(0, n) = 0 \quad \text{and} \quad \bar{V}(m, n) = \left(\frac{n}{m+n}\right)m + \frac{1}{m+n}\sum_{j=0}^{m-1}\bar{V}(j, n).$$

The result now follows from the Ratio Lemma.

LEMMA 6. $\bar{O}(m, n) \leq \ln(1+m/n)$ *for* $n \geq 1$ *and* $\bar{O}(m, 0) \leq \ln(1+m)+1$.

PROOF. The probability of choosing an element in N is $n/(m+n)$ and the expectation of O conditional on this event is zero. The probability of choosing a fixed element c (which is $(j+1)$st in the ordering on M) is $1/(m+n)$ and expectation of O conditional on c being chosen is $1+\bar{O}(j, n)$ by the restriction argument. Thus we get the following

recurrence relation:

$$\bar{O}(0, n) = 0 \quad \text{and}$$

$$\bar{O}(m, n) = \frac{n}{m+n} \cdot 0 + \frac{1}{m+n} \sum_{j=0}^{m-1} (1 + \bar{O}(j, n))$$

$$= \frac{m}{m+n} + \frac{1}{m+n} \sum_{j=0}^{m-1} \bar{O}(j, n).$$

If $n > 0$, the result follows from the Ratio Lemma. If $n = 0$, this means the experiment proceeds all the way until all elements from $M \cup N = M$ have been chosen. As $\bar{O}(0, 0) = 0$, we get $\bar{O}(m, 0) = 1 + 1/m \sum_{j=1}^{m-1} \bar{O}(j, 0)$, for $m > 0$. One can easily prove inductively that $\bar{O}(m, 0) \leq 1 + \ln m$, for $m \geq 1$.

Now let us extend our experiment a bit further. Assume $n \geq 2$. Let us keep on drawing elements from $M \cup N$ without replacement, until two elements from N are chosen. Let W denote the number of visible elements of M at the end of the experiment. Let $\bar{W}(m, n)$ be its expected value. We shall show next that for a constant $n \geq 2$, $\bar{W}(m, n)$ converges to a constant. In fact, what we shall really prove is a stronger result.

LEMMA 7. *$\bar{W}(m, n) \leq n$. Thus the bound does not depend on m and it depends linearly on n.*

PROOF. The probability of drawing an element from N is $n/(m+n)$ and the expectation of W conditional on this event is, by Lemma 5, $\leq (n-1) \ln(m/(n-1)+1)$. ($n-1$ appears because after drawing one element from N, $(n-1)$ remain.) The probability of drawing a fixed element c (which is $(j+1)$st in the ordering of M) is $1/(m+n)$ and the expectation of W conditional on this event is, by the restriction argument, $\bar{W}(j, n)$. Thus we get the following recurrence relation:

$$\bar{W}(m, n) \leq \frac{n}{m+n} (n-1) \ln\left(\frac{m}{n-1} + 1\right) + \frac{1}{m+n} \sum_{j=0}^{m-1} \bar{W}(j, n)$$

$$\leq \frac{1}{\frac{m}{n} + 1} (n-1)\left(\ln\left(\frac{m}{n} + 1\right) + \ln \frac{n}{n-1}\right) + \frac{1}{m+n} \sum_{j=0}^{m-1} \bar{W}(j, n).$$

Using the Unit Convergence Lemma, we conclude

$$\bar{W}(m, n) \leq (n-1)\left(1 + \ln\left(1 + \frac{1}{n-1}\right)\right) \leq (n-1)\left(1 + \frac{1}{n-1}\right) = n.$$

4.3. PLAYING WITH SEGMENTS

Now we shall play another game, this time with segments in the plane. Suppose we have l segments on the plane. Consider two completely imaginary intersecting segments S, T in the plane. There could be some segments in the plane which intersect both S and T. Without loss of generality, we shall assume that S ends on T; if it does not, we can split S at its point of intersection with T and apply the following analysis separately to its two halves. Imagine an observer o sitting at the point of intersection of S and T. We assume that the observer can see only along S. We also assume that there is a distinguished point t on T.

Now we play a game. At the start of the game there is nothing on the plane, except the two imaginary segments S and T and the observer o. Now from a set of given segments we select a random segment and put it on the plane at its given place. Note that we are *not* throwing S at a random place on the plane. We are assuming that all the given segments have already been completely specified on the plane, i.e. to say no assumption about the spatial distribution of the given segments in the plane is made. Only the order of drawing a segment is random. We keep on successively selecting a segment randomly from the set of remaining segments, and drawing it on the plane as specified, until we draw a segment which intersects T in the interval $[o, t]$, at which point the game ends. Let $a_1, \ldots, a_k$ be the segments drawn (in that order) before the game ended. We say that a_i was observed by o if there is no a_j $(j < i)$ drawn previously which obscures the point of intersection of a_i and S from o.

For example, in Figure 4, a_1, a_2 were observed, others were not. In this example the game ended when a_5 was chosen. Let O be the number segments observed by the observer. Let a_r be the chosen segment intersecting S which is closest to o. The interval of S between o and the point of intersection of S and a_r is called the visible span at the end of the experiment. In Figure 4 the visible span is the interval between o and the point of intersection between S and a_2. Let V be the number of *remaining* segments (i.e. the segments that remained to be selected at the end of the experiment) which intersect the visible span. Let N be the set of given segments intersecting T in the interval $[o, t]$, and let n be its size. Let M be the set of given segments intersecting S but which are not in N, and let m be its size. By the restriction argument, the expected values of V and O can depend only on the sets M and N, and not on l, the total number of segments on the plane. Hence we can denote these expected values by $\bar{V}(m, n)$ and $\bar{O}(m, n)$ respectively.

LEMMA 8.

$$\bar{V}(m, 0) = 0,$$

$$\bar{V}(m, n) \leq n\left[\ln\left(1+\frac{m}{n}\right)+1\right] \quad \textit{for } n \geq 1,$$

$$\bar{O}(m, 0) \leq \ln(1+m)+2,$$

$$\bar{O}(m, n) \leq \left[\ln\left(1+\frac{m}{n}\right)+1\right] \quad \textit{for } n \geq 1.$$

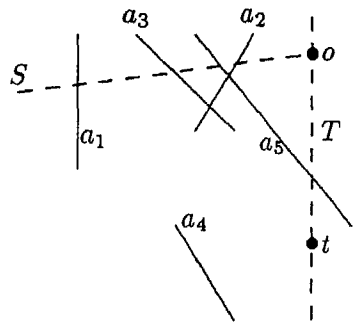


Figure 4

PROOF. If there were no segments which intersected both S and T then the lemma is a direct consequence of Lemma 5 and Lemma 6. But if there are segments which intersect both S and T, we argue as follows.

Note that the experiment automatically ends when one selects a segment which intersects both S and T. Hence the expected number of observed segments could be at the most one more than the bound obtained by applying Lemma 6 to the sets M and N. We get

$$\bar{O}(m, n) \leq \ln\left(1+\frac{m}{n}\right)+1, \qquad n \geq 1,$$

$$\bar{O}(m, 0) \leq \ln(1+m)+2.$$

By a similar argument the expected number of remaining segments intersecting the visible span is at the most n greater than the bound obtained by applying Lemma 5 to the sets M and N. Hence

$$\bar{V}(m, n) \leq n\left[\ln\left(1+\frac{m}{n}\right)+1\right],$$

$$\bar{V}(m, 0) = 0.$$

Let the segment S not be imaginary anymore. Instead, S will be one of the given segments which intersect T. Without loss of generality, we shall assume, as before, that S ends on T. The observer o sits at the intersection of S and T (see Figure 5). The sets N and M are defined as before and their sizes are m and n. Note that the segment S now belongs to N, but it does not belong to M. This time we shall draw the input segments at random all the way until each of them is drawn. But this time the observer will not be active throughout the game.

The observer remains inactive throughout the experiment if a segment in $N-\{S\}$ were drawn before S. Otherwise S is the first segment from N to be drawn. In this case the observer becomes active when S is drawn. He becomes inactive the moment a segment in $N-\{S\}$ is drawn; if $N-\{S\}$ is empty he remains active thereafter. Let O_S be the number of segments observed by the observer o during his active phase. By the restriction argument we can confine our attention to $M \cup N$ and hence the expected value of O_S can be bounded only in terms of m and n. A crucial result is that this expected value, $\bar{O}_S(m, n)$, can be bounded by a quantity which does not depend on m at all and depends inversely on n.

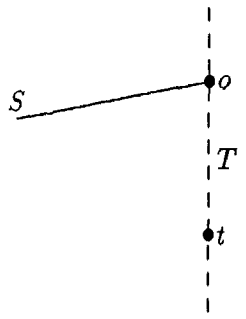


Figure 5

LEMMA 9. $\bar{O}_S(m, n) \leq (2/n) + O(1/n^2)$, *for* $n \geq 1$.

PROOF. By the restriction argument, we can restrict ourselves to the set $M \cup N$. The probability of drawing a fixed element c($(j+1)$st in order) in M is $1/(m+n)$ and, by restriction argument again, the conditional expectation of O_S is bounded by $\bar{O}_S(j, n)$. The probability of drawing S is $1/(m+n)$ and, by Lemma 8, the resulting conditional expectation is bounded by $\ln(m/(n-1)+1)+1$ if $n>1$. (If $n=1$, it is bounded by $\ln(m+1)+2$.) Assume first that $n>1$. The probability of drawing an element in $N-\{S\}$ is $n-1/(m+n)$ and the resulting conditional expectation is 0. Hence we can write for $\bar{O}_S(m, n)$ the following recurrence relation:

$$\bar{O}_S(m, n) \leq \frac{\ln\left(\frac{m}{n-1}+1\right)+1}{m+n} + \frac{1}{m+n}\sum_{j=0}^{m-1} \bar{O}_S(j, n) + \frac{n-1}{m+n} \cdot 0$$

$$\leq \frac{\ln\left(\frac{m}{n}+1\right)+1+\ln\frac{n}{n-1}}{m+n} + \frac{1}{m+n}\sum_{j=0}^{m-1} \bar{O}_S(j, n).$$

By the Unit Convergence Lemma we conclude that

$$\bar{O}_S(m, n) \leq \frac{1+1+\ln\frac{n}{n-1}}{n} = \frac{2}{n} + O\left(\frac{1}{n^2}\right).$$

If $n=1$, i.e. if S is the only element of N, we get the recurrence

$$\bar{O}_S(m, n) \leq \frac{\ln(m+1)+2}{m+1} + \frac{1}{m+1}\sum_{j=0}^{m-1} \bar{O}_S(j, n).$$

Again, by the Unit Convergence Lemma, we conclude that $\bar{O}_S(m, n) \leq 1+2=3$.

Now in the same game, let us say that we have a distinguished element R in the set N. The way this game is used will be such that the distinguished point t on T will actually be the point of intersection of R and T (see Figure 6).

Let us alter the active phase of o a little. Now o can become active at the time S is drawn only if (1) R has already been drawn and no element of $N-\{R, S\}$ has been drawn so far. The observer can also become active at the time R is drawn if (1) S has

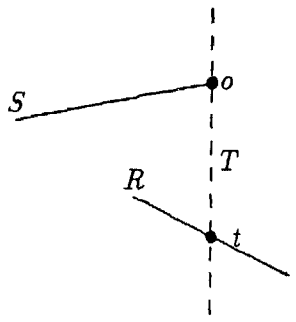


Figure 6

already been drawn and no element in $N-\{R, S\}$ has been drawn. If the above two situations do not occur o remains inactive. The above two rules can be combined into one rule by saying that o become active the moment *both* R and S have been drawn, provided no element in $N-\{R, S\}$ has been drawn so far. In this case (i.e. if o becomes active at all) o becomes inactive the moment some element of $N-\{R, S\}$ is drawn; if $N-\{R, S\}$ is empty, he remains active thereafter. Let O_R^S be the number of segments drawn which were observed by o in his active phase. Let $\bar{O}_R^S(m, n)$ be the expectation of O_R^S.

LEMMA 10. *$\bar{O}_R^S(m, n)$ is $O(1/n^2)$.*

PROOF. Again without loss of generality we can assume that S ends on T. The probability of drawing an element $c((j+1)$st in order) from M is $1/(m+n)$ and the conditional expectation of O_R^S is $\bar{O}_R^S(j, n)$. The probability of drawing R is $1/(m+n)$ and the conditional expectation of O_R^S is bounded by $a/(n-1)$ for some $a>0$ by Lemma 9. The probability of drawing S is $1/(m+n)$ the conditional expectation is again bounded by $a/(n-1)$. The probability of drawing an element from $N-\{R, S\}$ is $(n-2)/(m+n)$ and the conditional expectation is 0. Thus we get the recurrence relation

$$\bar{O}_R^S(0, n)=0 \quad \text{and} \quad \bar{O}_R^S(m, n)\leq\frac{2a/(n-1)}{m+n}+\frac{1}{m+n}\sum_{j=0}^{m-1}\bar{O}_R^S(j, n).$$

By the Unit Convergence Lemma,

$$\bar{O}_R^S(m, n)\leq\frac{2a}{n(n-1)}=O\left(\frac{1}{n^2}\right).$$

Finally, let us assume that the set N has two distinguished elements R, R' (see Figure 7).

The way this experiment will be used will be such that R and R' will be intersecting on T and t will be this shared point of intersection. The observer o will now become active (if at all) when all S, R, R' are drawn if no segment in $N-\{R, R', S\}$ has been drawn already. In case o becomes active, he becomes inactive the moment some segment in $N-\{R, R', S\}$ has been drawn. If $N-\{R, R', S\}$ is empty he remains active thereafter. Let $O_{R,R'}^S$ denote the number of segments observed by o during his active phase. Then its expected value is bounded as below.

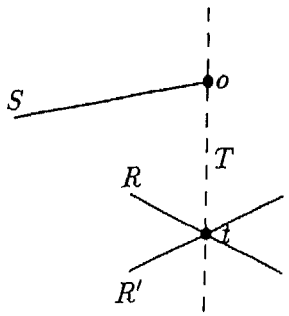


Figure 7

LEMMA 11. $\bar{O}^S_{RR'}(m, n)$ *is* $O(1/n^3)$.

PROOF. By using the above theorem and a similar argument we get the recurrence relation

$$\bar{O}^S_{RR'}(m, n) \leq \frac{3}{m+n} \cdot \frac{c}{n^2} + \frac{1}{m+n} \sum_{j=0}^{m-1} \bar{O}^S_{RR'}(j, n), \qquad \bar{O}^S_{RR'}(0, n) = 0.$$

The lemma follows.

We have assumed so far that the segment S ends on T. Now let us assume that S intersects T but does not end on T. The observer o sitting at the point of intersection can see along S in both directions. The random variables O_S, O^S_R, and $O^S_{RR'}$ can be appropriately defined in this case. Split S at the point of intersection, and apply the above analysis separately to its two halves. We thus get the bound on the expected values of these random variables by simply doubling the corresponding bounds in Lemma 9, Lemma 10, Lemma 11.

We are ready for the next game, again with segments in the plane. We are, as before, given a set of segments in the plane. And this time let us imagine an infinite imaginary vertical line T through the point of intersection of two fixed segments A, B from the given set of segments. We play the game of randomly drawing a segment as before. But this time our stopping rule will be different. We shall stop the moment both A and B have been drawn. Let us put an observer at t, the point of intersection of A and B. The observer can see only along T. Let V_{AB} be the number of remaining segments intersecting the visible span at the end of the experiment (the visible span is defined as before). Let n_{AB} be the number of given segments which intersect the line T. By the restriction argument, the expected value of V_{AB} can depend only on n_{AB}. But as the lemma below shows, this expected value can be bounded by a constant.

LEMMA 12. $E(V_{AB}) = \bar{V}_{AB}(n_{AB}) \leq 4$.

PROOF. Let T' be a semi-infinite starting at the point of intersection of A and B. If V'_{AB} is defined with respect to T', it suffices to prove that $E(V'_{AB}) \leq 2$.

To prove this let M be the set of segments intersecting T' other than A and B, and let N be the set consisting of A and B. The result follows by applying Lemma 7 to these sets.

4.4. FINISHING THE ANALYSIS

PROOF OF THEOREM 2. Let us denote the total number of face splits by F. Fix an endpoint t of an input segment. The vertical attachment through t keeps on contracting throughout the algorithm. Let F_t denote the number of times it is contracted during the algorithm. It is clear that $F = m + \sum_t F_t$. By the linearity of expectation, $E(F) = m + \sum_t E(F_t)$. It suffices to bound $E(F_t)$ for a fixed t. Let F_u (and F_l) be the number of times the upper (resp. lower) half of the contractible attachment through t is shrunk. By symmetry, it suffices to bound $E(F_u)$. Let m_u be the size of the set M_u of input segments which intersect the semi-infinite vertical line through t going upwards. We define M_l and m_l similarly.

Then in Lemma 6 if we let $M = M_u$ and $N =$ empty set, it is clear that $F_u = O$, where O is the random variable as defined in Lemma 6. Hence

$$\begin{aligned} E(F_t) = E(F_u) + E(F_l) &\leq (\ln(1 + m_u) + 1) + (\ln(1 + m_l) + 1) \\ &\leq 2\,(\ln(1 + m_t) + 1), \end{aligned}$$

where $m_t = m_u + m_l$. Letting t range over the n endpoints of input segments,

$$E(F) = m + \sum_t E(F_t) \leq m + 2\sum_t (\ln(1+m_t)+1)$$

$$\leq m + 2n\left(\ln\left(1+\left(\sum_t m_t\right)\Big/ n\right)+1\right)$$

$$= m + 2n(\ln(1+s)+1),$$

where $s = (\sum_t m_t)/n$ is the average span length of the input.

PROOF OF THEOREM 4. If t is a point of intersection, let us define in a similar fashion F_t as the number of times the vertical attachment through t was shrunk after it came into existence. If F is the total number of face splits, we have $F = m + \sum_t F_t$, where t will now range over the endpoints of the input segments as well as the points intersections of the input segments. When t is an endpoint of an input segment, one can easily see that the estimate for $E(F_t)$ from the proof of Theorem 2 still holds. Hence, we need to estimate $E(F_t)$ when t is a point of intersection of two input segments, say A and B. Fix t. Let T be the imaginary vertical line through t. Put an observer at t who can see only along T in both directions. The contractible vertical attachment through t comes into existence when both A and B have been selected. The extent of this attachment is precisely the visible span w.r.t. the observer at this moment. Hence, using the terminology of Lemma 12, we know that $F_t \leq V_{AB}$ and $E(F_t) \leq E(V_{AB}) \leq 4$. There are m points of intersection. Hence we get that $E(F) \leq 4m + 2n(\ln(1+s)+1)$, where n is the number of input segments and s is the average span length of the input.

PROOF OF THEOREM 3. As we noted before the statement of the theorem, we need only worry about the face transitions across input segments. In a face transition across a segment S we always chose to go left and turn around at the first vertex which is visible on both sides of S. We could have gone to our right as well. We shall visit different points of attachment depending on our choice of direction. We will say that a point of attachment witnesses a face transition if it could potentially be visited during the transition by either of these two ways. Let h be the number of points of attachment which witnessed a particular face transition. Let Q be the sum of h over all such face transitions. The total number of points of attachment visited during the face transitions in the whole course of the algorithm is obviously bounded by Q. (Heuristically, this number should be approximately half of Q, but that is not a theoretical guarantee.) Consider a vertical attachment through an endpoint t of some input segment. Let Q_t be the total number of times either end of this attachment witnessed a face transition. Then it is clear that $Q = \sum_t Q_t$, and hence it suffices to bound $E(Q_t)$.

Fix t for the rest of the analysis. Let T be the infinite vertical line through t. Let ϕ_t'' (and ϕ_t') be the set of input segments which intersect T above (resp. below) t, and set $\phi_t = \phi_t'' \cup \phi_t'$. Let n_t'', n_t', n_t be the sizes of the sets $\phi_t'', \phi_t', \phi_t$ respectively. For every $S \in \phi_t$, place an observer o_S at the intersection of S and T who can see along S in both directions. Let n_s be the number of input segments which intersect T between S and t (including S itself). The active phase of the observer o_S is defined as in Lemma 9. Thus o_s becomes active (if at all) at the moment S is selected provided no input segment in ϕ, whose point of intersection with T lies between S and t, has been selected before. He becomes inactive the moment such a segment is selected; if there is no such segment he

remains active thereafter. Define the random variable O_S as in Lemma 9. The only difference is that in Lemma 9 we assumed S ends on T, hence the bound given in that lemma will have to be doubled. *An endpoint of the vertical attachment through t can witness a face transition across an input segment S iff* (1) $S \in \phi_t$, (2) *the observer o_S is active at the moment of the transition, and* (3) *o_S observes during this transition the segment being added to the partition.* It follows that $Q_t = \sum_{S \in \phi_t} O_S$, and

$$E(Q_t) = \sum_{S \in \phi_t} E(O_S) = \sum_{S \in \phi_t^u} E(O_S) + \sum_{S \in \phi_t^l} E(O_S).$$

But, using the estimate of Lemma 9 and doubling it for the reason given above, we get:

$$\sum_{S \in \phi_t^u} E(O_S) \leq \sum_{S \in \phi_t^u} \frac{4}{n_S} + O\left(\frac{1}{n_S^2}\right) = \sum_{k=1}^{n_t^u} \frac{4}{k} + O\left(\frac{1}{k^2}\right) \approx 4 \ln n_t^u + c,$$

for some $c > 0$. An explicit calculation yields that $c \approx 4\gamma + \pi^2/3$, where γ is the Euler's constant. Using the same estimate for $\sum_{S \in \phi_t^l} E(O_S)$, we conclude that $E(Q_t) \leq 4 \ln n_t^u + 4 \ln n_t^l + 2c \leq 8 \ln(n_t/2) + 2c \leq 8(1 + \ln n_t)$. And

$$E(Q) = \sum_t E(Q_t) \leq \sum_t 8(1 + \ln n_t) \leq 8n\left(1 + \ln \frac{\sum_t n_t}{n}\right) = 8n(1 + \ln s),$$

where s is the average span length of the input. Hence the total number of points of attachments visited in face transitions is bounded by $8n(1 + \ln s)$. (Realistically, as we remarked before, the number of points of attachment visited should be half of Q. Hence the constant will be closer to four.)

PROOF OF THEOREM 5. For a point of intersection t we define, in a similar fashion, Q_t as the number of face transitions witnessed by either end of the vertical attachment through t *after* it came into existence. As in Theorem 3, we see that the total number of points of attachments visited will be bounded by $Q = \sum_t Q_t$, where t will now range over the endpoints of input segments as well as the points of intersections. The bound on $E(Q_t)$ when t is an endpoint of an input segment is the same as in Theorem 3. We also need to estimate $E(Q_t)$, if t is a point of intersection of say R and R'. The set ϕ_t is defined analogously. We place an observer o_S at the intersection of S and an imaginary vertical line T through t. We define the active phase of this observer and the random variable $O^S_{RR'}$ as in Lemma 11, save for a minor change that S does not end on T as assumed in the lemma. Next one sees that $Q_t = \sum_{S \in \phi'} O^S_{RR'}$. Using the estimate of the lemma, it follows that

$$E(Q_t) = O\left(\sum_{S \in \phi_t} \frac{1}{n_S^3}\right) = O(1).$$

As t varies over all points of intersection, $E(Q_t)$ sums to $O(m)$, where m is the number of intersections. As in Theorem 3, when t varies over the endpoints of input segments $E(Q_t)$ sums to $O(n \ln s)$, where n is the number of input segments and s is the average span length of the input. Hence $E(Q)$ is $O(m + n \ln s)$.

4.5. WORST CASE ANALYSIS

Like Quicksort, randomization is essential to the efficiency of this algorithm, because the worst case complexity of both algorithms is $O(n^2 \alpha(n))$, where α is the inverse of Ackerman's function. We shall prove this for the second theorem, the proof for the first being essentially the same.

Consider the $(k+1)$th refinement step of the algorithm, when $S = S_{k+1}$ is being added to the partition G_k. It obviously suffices to prove that the number of vertices of G_k visited in this refinement is $O(n + k\alpha(k))$. We need only consider the points of intersection, and the points of attachment of the vertical attachments through them, because there are only $O(n)$ of the remaining vertices. Consider a partition G'_k obtained from G_k by erasing all vertical attachments and removing from each S_i, $i \leq k$, a very small open neighbourhood centered around the point of intersection of S_i with S. The faces of G'_k need not be simple or convex. Let $W = W_k$ be the face of G'_k containing S. If a point of intersection v was visited in the $(k+1)$th refinement step, it must lie on ∂W. But a little thought tells us that the same holds if a point of attachment of the vertical attachment through v was visited in a face transition of this refinement. It follows from Sharir *et al.* (1986), that the length of ∂W is $O(k\alpha(k))$. Hence, the cost of the $(k+1)$th refinement is $O(n + k\alpha(k))$, as claimed.

5. Degeneracy

Let us now remove our non-degeneracy assumption that distinct endpoints have distinct x-coordinates. A special case of this occurs when two or more input segments share an endpoint. This can hardly be called a degeneracy, because it is the most common occurrence. In this case, we form the initial partition G_0 by passing just one vertical attachment through the shared endpoint. The rest of the algorithm remains essentially the same, save for some minor modifications. The other case of degeneracy occurs when two distinct endpoints have the same x coordinate. This too is quite common in the computer graphics context. In this case, one forms the initial partition by passing *distinct* vertical attachments through these endpoints. This creates strips of zero area, but the algorithm can handle that easily.

There are other rare degeneracies such as three segments sharing a point of intersection. Theoretically, all such degeneracies can be handled by just "perturbing" the input slightly. In practice, this is costly. In the actual implementation of the algorithm, we were able to handle all these degeneracies in a uniform way by following a few "consistency" rules, and then proving that the resulting constrained algorithm always works correctly. As the proof of correctness is rather tedious, we shall not discuss these degeneracies in any further detail here.

It is my pleasure to thank Janos Simon and Mike Wichura for helpful discussions.

Note added in proof: A new, simpler analysis of the algorithm in this paper is given in Mulmuley (1989*a*).

References

Bentley, J. L., Ottmann, T. (1979). Algorithms for reporting and counting geometric intersections. *IEEE Trans. on Computers* **C28**, 643–647.

Chazelle, B. (1986). Reporting and counting segment intersections. *J. Comput. Sys. Sci.* **32**, 156–182.

Chazelle, B., Edelsbrunner, H. (1988). An optimal algorithm for intersecting line segments in the plane. *Proc. of the 29th Annual IEEE Symposium on Foundations of Computer Science* 590–600.

Clarkson, K. (1988). Applications of random sampling in computational geometry, II, *Proc. of the 4th Annual ACM Symposium on Computational Geometry* 1–11.

Edelsbrunner, H., O'Rourke, J., Seidel, R. (1986). Constructing arrangements of lines and hyperplanes with applications. *SIAM J. Computing* **15**, 341–363.

Mulmuley, K. (1989*a*). A fast planar partition algorithm, II. *Proc. of the 5th Annual ACM Symposium on Computational Geometry* 33–43.

Mulmuley, K. (1989*b*). An efficient algorithm for hidden surface removal. *Proc. of the ACM SIGGRAPH, Computer Graphics*, **23**(3), 379–388.

Sharir, *et al.* (1986). Geometric applications of Davenport-Schinzel sequences. *Proc. of the 27th Annual IEEE Symposium on Foundations of Computer Science* 77–86.

Sutherland, I. E., Sproull, R. F., Schumaker, R. A. (1974). A characterization of ten hidden surface removal algorithms. *Comput. Surv.* **6**, 1–55.